NVIDIA Ampere vGPU 配置与测试参考

修订记录

Date	Revision	Editor	Changes
2020/11/12	1.0	Merlin Ma	*Initial version.*
2020/11/16	1.1	Merrin Ma	*Some typo corrections and CI testing.*
2021/08/05	2.0	Merlin Ma	*For Ampere GPU models (not only A100).* *None-MIG SR-IOV vGPU configuration added. Document restructuring. (chapter 4)*
2021/08/10	2.1	Merlin Ma	*Detailed SR-IOV GPU explanation,rules(chapter 2.4, 2.5).* *More examples.*

概述

NVidia A100 GPU 提供了全新的 MIG 多实例 GPU 模式，MIG 功能可以将 GPU 安全地划分成为最多 7 个独立地 GPU 计算实例，每个计算实例拥有自己所属的计算资源，提供更好的 QoS 以及故障隔离能力。从 vGPU 11.1 版本开始支持基于 MIG 技术的 vGPU 虚拟机实例。本文将着重介绍: 使用 NVidia A100 GPU 在 MIG 模式下 vGPU 实例的创建和管理，主要面向 MIG vGPU 的 KVM 虚拟化环境 PoC 测试和部署方法参考。同时，本文也以 A100 GPU 为例，介绍了基于 SR-IOV 的 GPU 的 vGPU 配置方法，也适用于 A10、A40、A16 等 GPU 的时分 vGPU 方案。

1. 安装需求和准备工作

1. 主机配置: NVidia Ampere GPU 的 X86_64 服务器。
2. KVM HOST: 安装 Redhat RHEL 8.2 及其以上版本。
3. GRID 11.1 或者以上版本试用软件及许可证，请先确认 vGPU 软件与物理 GPU 的兼容性。

2. KVM Host 软硬件基础配置概要

2.1. 修改 Linux kernel 启动参数

在 KVM 平台的 Linux kernel 启动项中添加 intel_iommu=on iommu=pt
通常此配置文件在：/boot/efi/EFI/redhat/grubenv 或 /boot/grub2/grubenv

这是修改后的 RHEL8.2 kenel 参数: (添加蓝色部分):

```
[root@kvm ~]# cat /boot/grub2/grubenv
# GRUB Environment Block
saved_entry=11f957e144e643c68255fd47642d77b3-4.18.0-193.el8.x86_64
kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lv
m.lv=rhel/swap rhgb intel_iommu=on iommu=pt quiet
boot_success=0
boot_indeterminate=0
################################################################################################################
################################################################################################################
################################################################################################################
################################################################################################################
################################################################################################################
################################################################################################################
################################################################################
[root@kvm ~]# cat /proc/cmdline
BOOT_IMAGE=(hd2,gpt2)/vmlinuz-4.18.0-193.el8.x86_64 root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/m
apper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb intel_iommu=on iommu=pt quiet
[root@kvm ~]#
```

2.2. 安装 vGPU 的 KVM Host 驱动

rpm -ivh NVIDIA-vGPU-rhel-8.2-450.89.x86_64.rpm。

```
[root@kvm Host]# ls -l NVIDIA-vGPU-rhel-8.2-450.89.x86_64.rpm
-rw-rw-r-- 1 root root 18239080 Nov  5 01:13 NVIDIA-vGPU-rhel-8.2-450.89.x86_64.rpm
[root@kvm Host]# rpm -ivh NVIDIA-vGPU-rhel-8.2-450.89.x86_64.rpm
Verifying...                          ################################# [100%]
Preparing...                          ################################# [100%]
Updating / installing...
   1:NVIDIA-vGPU-rhel-1:8.2-450.89    ################################# [100%]
[root@kvm Host]#
```

2.3. Reboot 并确认 GPU 状态

修改 kernel 参数和安装 vGPU Host 驱动以后，一定要 reboot 使其生效。

检查 nvidia-smi 输出: 应该看到 A100 GPU，默认状态 MIG 是被禁用状态的。
记录下 A100 GPU 的 PCI BUS ID，本例中为: **0000:81:00.0**

```
[root@kvm ~]# nvidia-smi
Thu Nov 12 23:17:19 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.89       Driver Version: 450.89       CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  A100-PCIE-40GB      On   | 00000000:81:00.0 Off |                    0 |
| N/A   29C    P0    35W / 250W |      0MiB / 40537MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            On   | 00000000:84:00.0 Off |                    0 |
| N/A   35C    P8    16W /  70W |     83MiB / 15359MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

2.4. 启用 SRIOV

注意：在执行此步骤之前，请确保物理 GPU 未被任何其他进程使用，例如: CUDA 程序、监控程序或 nvidia-smi 命令。仅使用随 NVIDIA vGPU 软件提供的自定义脚本 sriov-manage。不要尝试通过其他方式使能 SR-IOV。

运行/usr/lib/nvidia/sriov-manage -e 加上前面记下的 Bus-ID，例如：

/usr/lib/nvidia/sriov-manage -e 0:81:0.0

```
[root@kvm ~]# /usr/lib/nvidia/sriov-manage -e 0:81:0.0
Enabling VFs on 0000:81:00.0
[root@kvm ~]#
```

Kernel log 可以看到:

```
NVRM: GPU at 0000:81:00.0 has software scheduler ENABLED with policy VGPU_RELATIVE.
nvidia 0000:81:00.4: enabling device (0000 -> 0002)
nvidia 0000:81:00.4: MDEV: Registered
nvidia 0000:81:00.5: enabling device (0000 -> 0002)
nvidia 0000:81:00.5: MDEV: Registered
nvidia 0000:81:00.6: enabling device (0000 -> 0002)
nvidia 0000:81:00.6: MDEV: Registered
nvidia 0000:81:00.7: enabling device (0000 -> 0002)
nvidia 0000:81:00.7: MDEV: Registered
nvidia 0000:81:01.0: enabling device (0000 -> 0002)
nvidia 0000:81:01.0: MDEV: Registered
nvidia 0000:81:01.1: enabling device (0000 -> 0002)
nvidia 0000:81:01.1: MDEV: Registered
nvidia 0000:81:01.2: enabling device (0000 -> 0002)
nvidia 0000:81:01.2: MDEV: Registered
```

如果执行 SRIOV 使能时发生如下错误，需要检查 kernel 参数中的 iommu 选项是否生效。或服务器 BIOS 中的相关选项。

```
[root@kvm nvidia]# cd /usr/lib/nvidia/
[root@kvm nvidia]# ls
common.sh  post-install  pre-uninstall  sriov-manage  systemd  sysv  upstart
[root@kvm nvidia]# lspci |grep -i NV
81:00.0 3D controller: NVIDIA Corporation Device 20f1 (rev a1)
84:00.0 3D controller: NVIDIA Corporation TU104GL [Tesla T4] (rev a1)
[root@kvm nvidia]# ./sriov-manage -e 0:81:0.0
Enabling VFs on 0000:81:00.0
./sriov-manage: line 97: /sys/bus/pci/drivers/nvidia/bind: No such file or directory
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
./sriov-manage: line 142: echo: write error: No such device
```

看到 dmesg kenrel log 有报错:

```
328.670507] NVRM: Aborting probe for VF 0000:81:00.4 since IOMMU is not present on the system.
328.670514] nvidia: probe of 0000:81:00.4 failed with error -1
328.670668] NVRM: Aborting probe for VF 0000:81:00.5 since IOMMU is not present on the system.
328.670673] nvidia: probe of 0000:81:00.5 failed with error -1
328.670807] NVRM: Aborting probe for VF 0000:81:00.6 since IOMMU is not present on the system.
328.670811] nvidia: probe of 0000:81:00.6 failed with error -1
328.670941] NVRM: Aborting probe for VF 0000:81:00.7 since IOMMU is not present on the system.
328.670945] nvidia: probe of 0000:81:00.7 failed with error -1
328.671094] NVRM: Aborting probe for VF 0000:81:01.0 since IOMMU is not present on the system.
328.671129] nvidia: probe of 0000:81:01.0 failed with error -1
328.671306] NVRM: Aborting probe for VF 0000:81:01.1 since IOMMU is not present on the system.
328.671323] nvidia: probe of 0000:81:01.1 failed with error -1
```

确认启用了 SR-IOV 之后，对于 A100 和 A30 等支持 MIG 特性的物理 GPU，既可以使用常规 Time Sliced 类型的 vGPU，也可以使用 MIG 后端类型 vGPU。而 A10，A40 等不支持 MIG 特性的 GPU 则只支持 Time Sliced 方式的 vGPU.

通过 lspci 命令可查看 GPU 支持的 SR-IOV 属性，例如查看本案中 81:00.0 的 A100 GPU:

lspci -s 00:81:00.0 -vv

```
        Capabilities: [600 v1] Vendor Specific Information: ID=0001 Rev=1 Len=024 <?>
        Capabilities: [900 v1] #19
        Capabilities: [bb0 v1] #15
        Capabilities: [bcc v1] Single Root I/O Virtualization (SR-IOV)
                IOVCap: Migration-, Interrupt Message Number: 000
                IOVCtl: Enable+ Migration- Interrupt- MSE+ ARIHierarchy+
                IOVSta: Migration-
                Initial VFs: 16, Total VFs: 16, Number of VFs: 16, Function Dependency Link: 00
                VF offset: 4, stride: 1, Device ID: 20f1
                Supported Page Size: 00000573, System Page Size: 00000001
                Region 0: Memory at c9000000 (32-bit, non-prefetchable)
                Region 1: Memory at 0000025000000000 (64-bit, prefetchable)
                Region 3: Memory at 0000026002000000 (64-bit, prefetchable)
                VF Migration: offset: 00000000, BIR: 0
        Capabilities: [c14 v1] Alternative Routing-ID Interpretation (ARI)
                ARICap: MFVC- ACS-, Next Function: 0
                ARICtl: MFVC- ACS-, Function Group: 0
        Capabilities: [c1c v1] #26
        Capabilities: [d00 v1] #27
        Capabilities: [e00 v1] #25
        Kernel driver in use: nvidia
        Kernel modules: nouveau, nvidia_vgpu_vfio, nvidia
```

看到 A100 最大分配 VF 数量为 16 个。

所以，使能 SRIOV 后，在 /sys/class/mdev_bus/ 目录中可以列出所有可用于创建 vGPU 的 VF 设备的 BDF，A100 的 00:81:* 的 VF 目录一共是 **16** 个。

```
[root@kvm mdev_bus]# pwd
/sys/class/mdev_bus
[root@kvm mdev_bus]# ls
0000:81:00.4  0000:81:00.7  0000:81:01.2  0000:81:01.5  0000:81:02.0  0000:81:02.3
0000:81:00.5  0000:81:01.0  0000:81:01.3  0000:81:01.6  0000:81:02.1  0000:84:00.0
0000:81:00.6  0000:81:01.1  0000:81:01.4  0000:81:01.7  0000:81:02.2
[root@kvm mdev_bus]#
```

2.5. 基于 SRIOV 的 vGPU 特性

创建 SR-IOV 类型的 vGPU，遵循以下方法:

1. **每个 vGPU 实例占用一个 VF 设备，一旦 VF 已经被分配，该 VF 上不可再创建 vGPU。**
2. 每物理 GPU 的 VF 总数 >= 该 GPU 上可创建的 vGPU 最大实例数，例如 A100 单卡的 Total VF 数量为 16，而最大 A100 单卡的 vGPU 实例数为 40GB(FB 总量)/4GB(4C 类型 vGPU Size) = 10。因此，VF 总数并**不是**该 GPU 的可创建最大 vGPU 数量。
3. 最大可创建的 vGPU 实例数可以查询该 vGPU 类型目录中的 **description** 文件中描述的 **max_instances** 的值。例如:

```
[root@kvm 0000:81:00.4]# cd mdev_supported_types/
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.4/mdev_supported_types
[root@kvm mdev_supported_types]# cd nvidia-468/
[root@kvm nvidia-468]# cat name
GRID A100-4C
[root@kvm nvidia-468]# cat description
num_heads=1, frl_config=60, framebuffer=4096M, max_resolution=4096x2160, max_instance=10
[root@kvm nvidia-468]#
[root@kvm nvidia-468]#
```

4. 每一个 VF 设备的当前实时可创建 vGPU 数量应查询该 VF 目录下，指定类型目录中 **available_instances** 文件中的值：**1** 表示可以在此 VF 设备创建指定的 vGPU 类型实例。**0** 则表示不可创建此类型 vGPU，可能是由于当前 VF 已经被占用，或当前 vGPU 类型不被支持。

```
[root@kvm nvidia-468]# pwd
/sys/class/mdev_bus/0000:81:00.4/mdev_supported_types/nvidia-468
[root@kvm nvidia-468]# cat available_instances
1
[root@kvm nvidia-468]#
```

下面将分开介绍创建 MIG 和 Non-MIG 模式的 vGPU 实例

如果您使用的是 A10、A40、A16 等非 MIG GPU, 请直接跳至本文第 4 章。

3. MIG 类型的 vGPU 配置概要

如果您的物理 GPU 支持 MIG 特性，例如 A100，A30。则既可以使用 MIG 模式的 vGPU 以实现物理分区的实例隔离性，也可以使用 Time Sliced(时间片方式切分共享)模式，本章节介绍 MIG 模式的 vGPU 的配置。如果您使用时间片切分的 vGPU 请参考下一个章节。

3.1. 启用 MIG 模式

创建基于 MIG 的 vGPU 实例需要将 GPU 切换到 MIG 模式, 此配置为持久配置, 只需执行一次, 重启服务器后仍然有效:

```
nvidia-smi -mig  1
```

```
[root@kvm ~]# nvidia-smi -i 0 -mig 1
Enabled MIG Mode for GPU 00000000:81:00.0
All done.
[root@kvm ~]# nvidia-smi
Thu Nov 12 23:39:59 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.89       Driver Version: 450.89       CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  A100-PCIE-40GB      On   | 00000000:81:00.0 Off |                   On |
| N/A   40C    P0    87W / 250W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   1  Tesla T4            On   | 00000000:84:00.0 Off |                    0 |
| N/A   38C    P8    16W /  70W |     83MiB / 15359MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| MIG devices:                                                                |
+------------------+----------------------+-----------+----------------------+
| GPU  GI  CI  MIG |         Memory-Usage |        Vol|         Shared       |
|      ID  ID  Dev |           BAR1-Usage | SM     Unc| CE  ENC  DEC  OFA  JPG|
|                  |                      |        ECC|                      |
|==================+======================+===========+======================|
|  No MIG devices found                                                       |
+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3.2. MIG 设备初始化

SR-IOV GPU 的每一个 VF 设备目录下只能创建一个 vGPU 实例。这个在 A100 时分模式的 SRIOV based vGPU 同样适用。

这里我们可以检索每一个 VF 目录所能创建的 vGPU 类型和数量:

```
[root@kvm mdev_bus]# ls
0000:81:00.4  0000:81:00.7  0000:81:01.2  0000:81:01.5  0000:81:02.0  0000:81:02.3
0000:81:00.5  0000:81:01.0  0000:81:01.3  0000:81:01.6  0000:81:02.1  0000:84:00.0
0000:81:00.6  0000:81:01.1  0000:81:01.4  0000:81:01.7  0000:81:02.2
[root@kvm mdev_bus]# cd 0000\:81\:00.4/mdev_supported_types/
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
[root@kvm mdev_supported_types]# for i in * ; do echo "  " $(cat $i/name) available: $(cat $i/avai*); done
   GRID A100-4C available: 0
   GRID A100-5C available: 0
   GRID A100-8C available: 0
   GRID A100-10C available: 0
   GRID A100-20C available: 0
   GRID A100-40C available: 0
   GRID A100-1-5C available: 0
   GRID A100-2-10C available: 0
   GRID A100-3-20C available: 0
   GRID A100-4-20C available: 0
   GRID A100-7-40C available: 0
[root@kvm mdev_supported_types]#
```

可以看到如果不创建 MIG GI, 则所以的 MIG 类型的 vGPU 可创建数量均为 0，因此我们需要先为 vGPU 创建相应的 MIG GPU Instance.

有关 MIG 的管理，请参见:

https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf

3.3. 创建用于 vGPU 的 MIG GPU Instance (GI)

这里我们创建 2 个测试 GI，分别为 MIG 4g.20gb 和 MIG 2g.10gb

列出 MIG GI Profile:

```
[root@kvm ~]# nvidia-smi mig -lgip
+-----------------------------------------------------------------------------+
| GPU instance profiles:                                                      |
| GPU   Name          ID    Instances   Memory     P2P    SM    DEC   ENC   |
|                           Free/Total   GiB              CE    JPEG  OFA   |
|=============================================================================|
|   0  MIG 1g.5gb     19     7/7         4.75       No     14     0     0   |
|                                                          1     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 2g.10gb    14     3/3         9.75       No     28     1     0   |
|                                                          2     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 3g.20gb     9     2/2         19.62      No     42     2     0   |
|                                                          3     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 4g.20gb     5     1/1         19.62      No     56     2     0   |
|                                                          4     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 7g.40gb     0     1/1         39.50      No     98     5     0   |
|                                                          7     1     1   |
+-----------------------------------------------------------------------------+
```

成功创建 GI:

```
[root@kvm ~]# nvidia-smi mig -cgi 5,14
Successfully created GPU instance ID  1 on GPU  0 using profile MIG 4g.20gb (ID  5)
Successfully created GPU instance ID  5 on GPU  0 using profile MIG 2g.10gb (ID 14)
[root@kvm ~]# nvidia-smi mig -lgi
+-------------------------------------------------------+
| GPU instances:                                        |
| GPU   Name          Profile  Instance   Placement    |
|                       ID       ID       Start:Size   |
|=======================================================|
|   0  MIG 2g.10gb      14        5          4:2        |
+-------------------------------------------------------+
|   0  MIG 4g.20gb       5        1          0:4        |
+-------------------------------------------------------+
```

3.4. 创建 vGPU 设备

此时查看 vGPU profile 各类型中可创建的数量:

```
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.4/mdev_supported_types
[root@kvm mdev_supported_types]# for i in * ; do echo $i: $(cat $i/name) available: $(cat $i/avai*); done
nvidia-468: GRID A100-4C available: 0
nvidia-469: GRID A100-5C available: 0
nvidia-470: GRID A100-8C available: 0
nvidia-471: GRID A100-10C available: 0
nvidia-472: GRID A100-20C available: 0
nvidia-473: GRID A100-40C available: 0
nvidia-474: GRID A100-1-5C available: 0
nvidia-475: GRID A100-2-10C available: 1
nvidia-476: GRID A100-3-20C available: 0
nvidia-477: GRID A100-4-20C available: 1
nvidia-478: GRID A100-7-40C available: 0
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
```

即可对于 A100-2-10C，或者 A100-4-20C 创建 MIG vGPU 实例。

这里选择第一个 VF 目录 /sys/class/mdev_bus/0000:81:00.4/mdev_supported_types

分别对应目录 nvidia-475, nvidia-477。

```
[root@kvm mdev_supported_types]# cd nvidia-475/
[root@kvm nvidia-475]# ls
available_instances  create  description  device_api  devices  name
[root@kvm nvidia-475]# uuidgen > create
[root@kvm nvidia-475]# ls
available_instances  create  description  device_api  devices  name
[root@kvm nvidia-475]# ls devices
c9035e18-77ac-4b54-9df0-284ff30feece
[root@kvm nvidia-475]#
```

上面创建了一个 2-10C vGPU 设备，如果创建第二个，则需要到另一个 VF 目录，这里转到 /sys/class/mdev_bus/0000:81:00.5/mdev_supported_types。可以看到仅有 1 个 4-20C 的 vGPU 类型可以创建。

```
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.5/mdev_supported_types
[root@kvm mdev_supported_types]# for i in * ; do echo $i: $(cat $i/name) available: $(cat $i/avai*); done
nvidia-468: GRID A100-4C available: 0
nvidia-469: GRID A100-5C available: 0
nvidia-470: GRID A100-8C available: 0
nvidia-471: GRID A100-10C available: 0
nvidia-472: GRID A100-20C available: 0
nvidia-473: GRID A100-40C available: 0
nvidia-474: GRID A100-1-5C available: 0
nvidia-475: GRID A100-2-10C available: 0
nvidia-476: GRID A100-3-20C available: 0
nvidia-477: GRID A100-4-20C available: 1
nvidia-478: GRID A100-7-40C available: 0
[root@kvm mdev_supported_types]#
```

创建第二个 vGPU MDEV 设备

```
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.5/mdev_supported_types
[root@kvm mdev_supported_types]# uuidgen > nvidia-477/create
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
[root@kvm mdev_supported_types]# ls nvidia-477/devices/
af430b8a-cc42-4025-a5f2-bbbc31b59062
[root@kvm mdev_supported_types]#
```

查看全部创建的 vGPU 设备

```
[root@kvm ~]# cd /sys/bus/mdev/devices
[root@kvm devices]# ls
af430b8a-cc42-4025-a5f2-bbbc31b59062  c9035e18-77ac-4b54-9df0-284ff30feece
[root@kvm devices]# ls -l
total 0
lrwxrwxrwx 1 root root 0 Nov 13 00:25 af430b8a-cc42-4025-a5f2-bbbc31b59062 -> ../../../devices/pci0000:80/0000:80:02.0/0000:81:00.5/af430b8a-cc42-4025-a5f2-bbbc31b59062
lrwxrwxrwx 1 root root 0 Nov 13 00:17 c9035e18-77ac-4b54-9df0-284ff30feece -> ../../../devices/pci0000:80/0000:80:02.0/0000:81:00.4/c9035e18-77ac-4b54-9df0-284ff30feece
[root@kvm devices]#
```

3.5. 创建 VM，向 VM 添加 MIG vGPU 设备

1. **创建 VM，并添加 vGPU mdev 设备**
2. 此步与 Non-MIG 模式 vGPU 的配置相同, 这里请参见标准添加步骤。
   以下是 KVM 虚拟机中添加的 mdev 设备的片段。

```
<hostdev mode='subsystem' type='mdev' managed='no' model='vfio-pci' display='on'>
  <source>
    <address uuid='af430b8a-cc42-4025-a5f2-bbbc31b59062'/>
  </source>
  <alias name='hostdev0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
</hostdev>
```

3.6. VM 内安装 vGPU 客户机驱动程序和 CUDA

VM 启动后，需要在虚拟机内安装 vGPU Guest 驱动程序和 CUDA 框架。

```
Installing vGPU guest driver, please wait...
Verifying archive integrity... OK
Uncompressing NVIDIA Accelerated Graphics Driver for Linux-x86_64 450.89.....
.....
.....
.....
.....
.....
Fri Nov 13 00:45:18 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.89       Driver Version: 450.89       CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4-20C     On   | 00000000:00:0A.0 Off |                   On |
| N/A   N/A    P0    N/A /  N/A |   1844MiB / 20475MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| MIG devices:                                                                |
+------------------+----------------------+-----------+----------------------+
| GPU  GI  CI  MIG |         Memory-Usage |        Vol|         Shared       |
|      ID  ID  Dev |           BAR1-Usage | SM     Unc| CE  ENC  DEC  OFA  JPG|
|                  |                      |        ECC|                      |
|==================+======================+===========+======================|
|  No MIG devices found                                                       |
+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Installing CUDA, please wait...
Done.
Building gpu_burn...
[root@test000081005_4-20C_1 ~]#
```

A100 MIG 模式的 vGPU 需要 vCS license 授权

3.7. VM 内创建 MIG Compute Instance

MIG vGPU 需要创建计算实例才能运行 CUDA 计算:
也可以根据需要创建不同规格的 CI 实例，默认会创建最大 profile 的 MIG 计算实例。
nvidia-smi mig -cci

```
[root@test000081005_4-20C_1 ~]# nvidia-smi  mig -cci
Successfully created compute instance ID  0 on GPU  0 GPU instance ID  0 using profile MIG 4g.20gb (ID  3)
[root@test000081005_4-20C_1 ~]# nvidia-smi
Fri Nov 13 00:55:50 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.89       Driver Version: 450.89       CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4-20C     On   | 00000000:00:0A.0 Off |                   On |
| N/A   N/A    P0    N/A /  N/A |   1844MiB / 20475MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| MIG devices:                                                                |
+------------------+----------------------+-----------+----------------------+
| GPU  GI  CI  MIG |         Memory-Usage |        Vol|         Shared       |
|      ID  ID  Dev |           BAR1-Usage | SM     Unc| CE  ENC  DEC  OFA  JPG|
|                  |                      |        ECC|                      |
|==================+======================+===========+======================|
|  0    0   0   0  |   1844MiB / 20475MiB | 56      0 |  4   0    2    0    0 |
|                  |      0MiB /  4096MiB |           |                      |
+------------------+----------------------+-----------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3. CUDA 计算测试

```
[root@test000081005_4-20C_1 ~]# nvidia-smi  -L
GPU 0: GRID A100-4-20C (UUID: GPU-98420d02-2505-11eb-8d63-b7a74215b053)
  MIG 4g.20gb Device 0: (UUID: MIG-GPU-98420d02-2505-11eb-8d63-b7a74215b053/0/0)
[root@test000081005_4-20C_1 ~]# ls
bin  nvdrv-install.sh  NVIDIA_CUDA-11.0_Samples  NVIDIA-Linux-x86_64-450.89-grid.run  update_xpciid.sh
[root@test000081005_4-20C_1 ~]# cd bin/
[root@test000081005_4-20C_1 bin]# ls
compare.ptx  gpu_burn
[root@test000081005_4-20C_1 bin]# ./gpu_burn 20
GPU 0: GRID A100-4-20C (UUID: GPU-98420d02-2505-11eb-8d63-b7a74215b053)
  MIG 4g.20gb Device 0: (UUID: MIG-GPU-98420d02-2505-11eb-8d63-b7a74215b053/0/0)
Initialized device 0 with 20475 MB of memory (18183 MB available, using 16364 MB of it), using FLOATS
15.0%  proc'd: 1020 (5605 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:20 CST 2020

35.0%  proc'd: 3060 (8761 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:24 CST 2020

50.0%  proc'd: 4080 (8761 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:27 CST 2020

65.0%  proc'd: 6120 (8761 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:30 CST 2020

85.0%  proc'd: 8160 (8761 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:34 CST 2020

100.0%  proc'd: 9180 (8761 Gflop/s)   errors: 0   temps: --
        Summary at:   Fri Nov 13 00:59:37 CST 2020
```

看到运行 CUDA 计算正常。

3.8. 修改 MIG Compute Instance 并为不同的应用指定不同的 CI

- 列出当前分配的 CI:

  nvidia-smi -lci

```
[root@test000081005_4-20C_1 ~]# nvidia-smi mig -lci
+--------------------------------------------------------------------+
| Compute instances:                                                 |
| GPU     GPU       Name             Profile   Instance   Placement  |
|       Instance                       ID        ID       Start:Size |
|         ID                                                         |
|====================================================================|
|   0      0       MIG 4g.20gb          3         0          0:4     |
+--------------------------------------------------------------------+
```

- 删除不需要的 CI:

  nvidia-smi -dci -ci <CI_ID>

```
[root@test000081005_4-20C_1 ~]# nvidia-smi mig -dci -ci 0
Successfully destroyed compute instance ID  0 from GPU  0 GPU instance ID  0
```

- 列出支持的 CI 类型

  nvidia-smi -lcip

```
[root@test000081005_4-20C_1 ~]# nvidia-smi mig -lcip
+-----------------------------------------------------------------------------------------+
| Compute instance profiles:                                                              |
| GPU     GPU       Name             Profile  Instances   Exclusive       Shared       |
|       Instance                       ID     Free/Total     SM       DEC   ENC   OFA  |
|         ID                                                          CE    JPEG       |
|=========================================================================================|
|   0      0       MIG 1c.4g.20gb       0      4/4           14        2     0     0   |
|                                                                     4     0           |
+-----------------------------------------------------------------------------------------+
|   0      0       MIG 2c.4g.20gb       1      2/2           28        2     0     0   |
|                                                                     4     0           |
+-----------------------------------------------------------------------------------------+
|   0      0       MIG 4g.20gb          3*     1/1           56        2     0     0   |
|                                                                     4     0           |
+-----------------------------------------------------------------------------------------+
```

- 可以一次创建多个新的 CI

  nvidia-smi -cci <id1,id2,id3…>

  下面分别创建 2 个 1c 和一个 2c 实例，Profile ID 分别为 0，和 1。

```
[root@test000081005_4-20C_1 ~]# nvidia-smi mig -cci 1,0,0
Successfully created compute instance ID  0 on GPU  0 GPU instance ID  0 using profile MIG 2c.4g.20gb (ID  1)
Successfully created compute instance ID  1 on GPU  0 GPU instance ID  0 using profile MIG 1c.4g.20gb (ID  0)
Successfully created compute instance ID  2 on GPU  0 GPU instance ID  0 using profile MIG 1c.4g.20gb (ID  0)
[root@test000081005_4-20C_1 ~]#
```

- 设定 CUDA_VISIBLE_DEVICES 环境变量，运行 CUDA 程序。

```
[root@test000081005_4-20C_1 matrixMul]# nvidia-smi -L
GPU 0: GRID A100-4-20C (UUID: GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7)
  MIG 2c.4g.20gb Device 0: (UUID: MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/0)
  MIG 1c.4g.20gb Device 1: (UUID: MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/1)
  MIG 1c.4g.20gb Device 2: (UUID: MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/2)
[root@test000081005_4-20C_1 matrixMul]# CUDA_VISIBLE_DEVICES=MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/1 ./matrixMul
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Ampere" with compute capability 8.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 665.44 GFlop/s, Time= 0.197 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performancemeasurements. Results may vary when GPU Boost is enabled.
[root@test000081005_4-20C_1 matrixMul]#
```

```
[root@test000081005_4-20C_1 matrixMul]# CUDA_VISIBLE_DEVICES=MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/1 ./matrixMul
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Ampere" with compute capability 8.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 665.44 GFlop/s, Time= 0.197 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performancemeasurements. Results may vary when GPU Boost is enabled.
[root@test000081005_4-20C_1 matrixMul]# CUDA_VISIBLE_DEVICES=MIG-GPU-d5f222ad-27b7-11eb-b708-9dc2c5a19eb7/0/0 ./matrixMul
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Ampere" with compute capability 8.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 1219.78 GFlop/s, Time= 0.107 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS
```

通过上面测试，可以观察到 MIG 模式下的 vGPU VM 内通过 CI 划分，在应用间可以分配不同的资源，提供良好的 QoS 和灵活性。

4. Time Sliced 类型的 vGPU 配置概要

对于 Non-MIG 的 Ampere 架构 GPU，vGPU 仍然使用 Time Sliced(时分共享)模式。新架构启用了 SR-IOV，vGPU 的管理还是在 vfio-mdev 的基础之上的。SR-IOV 的 vGPU 设备分配和之前非 SR-IOV 的 mdev 设备创建和管理会有所不同。本章节将讲解基于 SR-IOV 的 vGPU (例如 A10，A40，A16)时分切分方式的配置方法。

4.1. 创建 vGPU 设备

这里我们仍然以 A100 GPU 为例，在启用 SR-IOV 之后，可以在 **/sys/class/mdev_bus** 目录中看到所有的 VF 设备目录，如下图。

```
[root@kvm mdev_bus]# cd /sys/class/mdev_bus/
[root@kvm mdev_bus]# ls
0000:81:00.4  0000:81:00.7  0000:81:01.2  0000:81:01.5  0000:81:02.0  0000:81:02.3
0000:81:00.5  0000:81:01.0  0000:81:01.3  0000:81:01.6  0000:81:02.1  0000:84:00.0
0000:81:00.6  0000:81:01.1  0000:81:01.4  0000:81:01.7  0000:81:02.2
[root@kvm mdev_bus]# ls 0000\:81\:00.4
ari_enabled              d3cold_allowed   iommu           mdev_supported_types  resource      subsystem
broken_parity_status     device           iommu_group     modalias              resource0     subsystem_device
class                    dma_mask_bits    irq             msi_bus               resource1     subsystem_vendor
config                   driver           local_cpulist   numa_node             resource1_wc  uevent
consistent_dma_mask_bits driver_override  local_cpus      physfn                resource3     vendor
current_link_speed       enable           max_link_speed  power                 resource3_wc
current_link_width       firmware_node    max_link_width  reset                 revision
[root@kvm mdev_bus]#
[root@kvm mdev_bus]#
```

注意：这里基于 SR-IOV 的 vGPU，每一个 VF 设备只对应一个 vGPU 实例。这是非 SRIOV 的 GPU 的主要区别。每个 VF 设备目录，例如上图中 0:81:00.4 VF 中只能创建一个 mdev 设备。同时 Time sliced 模式的 vGPU，仍然要遵循同物理设备的 vGPU 类型唯一的原则。

创建 vGPU 设备仍然是通过每个 VF 目录 /mdev_supported_types/*/create 来创建。

这里我们创建第一个 A100-**4C** 类型的 vGPU 实例:

1. 选择未使用的 VF 设备目录

   选择第一个设备目录：**0000:81:00.4**

   进入目录可以看到如下图的目录结构:

```
[root@kvm mdev_bus]#
[root@kvm mdev_bus]# cd 0000\:81\:00.4
[root@kvm 0000:81:00.4]# ls
ari_enabled              d3cold_allowed   iommu           mdev_supported_types  resource      subsystem
broken_parity_status     device           iommu_group     modalias              resource0     subsystem_device
class                    dma_mask_bits    irq             msi_bus               resource1     subsystem_vendor
config                   driver           local_cpulist   numa_node             resource1_wc  uevent
consistent_dma_mask_bits driver_override  local_cpus      physfn                resource3     vendor
current_link_speed       enable           max_link_speed  power                 resource3_wc
current_link_width       firmware_node    max_link_width  reset                 revision
[root@kvm 0000:81:00.4]# cd mdev_supported_types/
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
[root@kvm mdev_supported_types]#
```

2. **找出 4C 类型对应的目录**

下面脚本可以列出所有支持的 mdev 设备名称和 vGPU 设备可用数量

cd mdev_supported_types

*for i in * ; do echo $i, `cat $i/name` `cat $i/ava*` ; done*

```
[root@kvm mdev_supported_types]#
[root@kvm mdev_supported_types]# for  i in * ; do echo $i, `cat $i/name` `cat $i/ava*` ; done
nvidia-468, GRID A100-4C 1
nvidia-469, GRID A100-5C 1
nvidia-470, GRID A100-8C 1
nvidia-471, GRID A100-10C 1
nvidia-472, GRID A100-20C 1
nvidia-473, GRID A100-40C 1
nvidia-474, GRID A100-1-5C 0
nvidia-475, GRID A100-2-10C 0
nvidia-476, GRID A100-3-20C 0
nvidia-477, GRID A100-4-20C 0
nvidia-478, GRID A100-7-40C 0
[root@kvm mdev_supported_types]#
```

这里可用看到，A100-4C 类型的目录名为 nvidia-468，且可用实例数量为 1

所以可以在此 VF 上创建 1 个 4C 类型的 vGPU 实例，目录使用 **nvidia-468**

3. 使用 uuid 创建 vGPU 设备

uuidgen > nvidia-468/create

```
[root@kvm mdev_supported_types]# cd nvidia-468/
[root@kvm nvidia-468]# ls
available_instances  create  description  device_api  devices  name
[root@kvm nvidia-468]# uuidgen > create
[root@kvm nvidia-468]# ls devices
59b486f4-4205-438f-a24e-2ac2b5384a75
```

上图可以看到创建成功

```
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.4/mdev_supported_types
[root@kvm mdev_supported_types]#
[root@kvm mdev_supported_types]# for  i in * ; do echo $i, `cat $i/name` `cat $i/ava*` ; done
nvidia-468, GRID A100-4C 0
nvidia-469, GRID A100-5C 0
nvidia-470, GRID A100-8C 0
nvidia-471, GRID A100-10C 0
nvidia-472, GRID A100-20C 0
nvidia-473, GRID A100-40C 0
nvidia-474, GRID A100-1-5C 0
nvidia-475, GRID A100-2-10C 0
nvidia-476, GRID A100-3-20C 0
nvidia-477, GRID A100-4-20C 0
nvidia-478, GRID A100-7-40C 0
```

上图显示已分配 vGPU 的 VF 0:81:00.4 可用的剩余 vGPU 实例数，**全部为 0**。意味着当前 VF 已经不能再创建新的 vGPU 设备。**新的 vGPU 只能创建在不同的 VF 上。**

因此需要切换到另一个 VF 目录，例如：**0000:81:00.5**

检查可用 vGPU 实例状态:

```
[root@kvm 0000:81:00.5]# pwd
/sys/class/mdev_bus/0000:81:00.5
[root@kvm 0000:81:00.5]# cd mdev_supported_types/
[root@kvm mdev_supported_types]# for  i in * ; do echo $i, `cat $i/name` `cat $i/ava*` ; done
nvidia-468, GRID A100-4C 1
nvidia-469, GRID A100-5C 0
nvidia-470, GRID A100-8C 0
nvidia-471, GRID A100-10C 0
nvidia-472, GRID A100-20C 0
nvidia-473, GRID A100-40C 0
nvidia-474, GRID A100-1-5C 0
nvidia-475, GRID A100-2-10C 0
nvidia-476, GRID A100-3-20C 0
nvidia-477, GRID A100-4-20C 0
nvidia-478, GRID A100-7-40C 0
[root@kvm mdev_supported_types]#
```

可以看到空闲的 VF 上面仍然可以创建 1 个相同类型的 vGPU 实例。直到所有 VF 的可用 vGPU instance 全部为 0。

因此，下图在 VF 0:81:00.5 上成功创建了第二个 vGPU 设备。以此类推。

```
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.5/mdev_supported_types
[root@kvm mdev_supported_types]# ls
nvidia-468  nvidia-470  nvidia-472  nvidia-474  nvidia-476  nvidia-478
nvidia-469  nvidia-471  nvidia-473  nvidia-475  nvidia-477
[root@kvm mdev_supported_types]# pwd
/sys/class/mdev_bus/0000:81:00.5/mdev_supported_types
[root@kvm mdev_supported_types]# uuidgen > nvidia-468/create
[root@kvm mdev_supported_types]# ls nvidia-468/devices/
43805aeb-9d0a-4c11-bf66-83173eacb660
[root@kvm mdev_supported_types]#
```

4.2. 添加 vGPU MDEV 设备到 VM

后续步骤和标准的非 SR-IOV 的操作相同。

如下图，将之前生成 vGPU 的 uuid 写入虚拟机的 libvirt xml 文件。

```
[root@kvm mig-4-20c]# ls /sys/bus/mdev/devices/ -l
total 0
lrwxrwxrwx 1 root root 0 Jul 13 20:08 30af52db-81c2-487c-adfe-fb5ced0cf806 -> ../../..
0:03.0/0000:82:00.0/0000:83:08.0/0000:84:00.0/30af52db-81c2-487c-adfe-fb5ced0cf806
lrwxrwxrwx 1 root root 0 Aug  5 14:18 43805aeb-9d0a-4c11-bf66-83173eacb660 -> ../../..
0:02.0/0000:81:00.5/43805aeb-9d0a-4c11-bf66-83173eacb660
lrwxrwxrwx 1 root root 0 Aug  5 13:48 59b486f4-4205-438f-a24e-2ac2b5384a75 -> ../../..
0:02.0/0000:81:00.4/59b486f4-4205-438f-a24e-2ac2b5384a75
lrwxrwxrwx 1 root root 0 Jul 13 20:17 f88e4d52-87f7-4d71-86cf-8dd108617db3 -> ../../..
0:03.0/0000:82:00.0/0000:83:08.0/0000:84:00.0/f88e4d52-87f7-4d71-86cf-8dd108617db3
```

```
    <hostdev mode='subsystem' type='mdev' managed='no' model='vfio-pci' display='on'>
      <source>
        <address uuid='43805aeb-9d0a-4c11-bf66-83173eacb660'/>
      </source>
      <alias name='hostdev0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
    </hostdev>
```

之后启动 VM：

```
a100_000081005_4-20C_1 : ca989eb3-56e3-43f4-8e30-f64b0e93fc74
  Net:
    192.168.122.191/24  | VNET3 | 52:54:00:E2:36:01
  Disk:
    Type   Device   Target   Source
    ------------------------------------------------------------------------------------------------------------
    file   disk     vda      /var/lib/libvirt/images/a100_linked_1-ca989eb3-56e3-43f4-8e30-f64b0e93fc74.qcow2
```

4.3. 安装 vGPU Guest Driver

```
[root@a100_000081005_4-20C_1 ~]# ls
bin  nv-docker-install.sh  NVIDIA_CUDA-11.0_Samples  NVIDIA-Linux-x86_64-460.73.01-grid.run  perl5
[root@a100_000081005_4-20C_1 ~]# ./NVIDIA-Linux-x86_64-460.73.01-grid.run
Verifying archive integrity... OK
Uncompressing NVIDIA Accelerated Graphics Driver for Linux-x86_64 460.73.01.....................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................
[root@a100_000081005_4-20C_1 ~]#
```

安装完成

```
[root@a100_000081005_4-20C_1 ~]# nvidia-smi
Thu Aug  5 02:32:03 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.73.01    Driver Version: 460.73.01    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4C       On    | 00000000:00:0A.0 Off |                    0 |
| N/A   N/A    P0    N/A /  N/A |    407MiB /  4091MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[root@a100_000081005_4-20C_1 ~]#
```

nvidia-smi 正常显示信息

```
[root@a100_000081005_4-20C_1 ~]# cd bin
[root@a100_000081005_4-20C_1 bin]# ls
compare.ptx  gpu_burn
[root@a100_000081005_4-20C_1 bin]# ./gpu_burn 5
GPU 0: GRID A100-4C (UUID: GPU-01bd2cb9-f5b5-11eb-87db-9d30a27a9dc2)
Initialized device 0 with 4091 MB of memory (3012 MB available, using 2711 MB of it), using FLOATS
20.0%  proc'd: 167 (2446 Gflop/s)   errors: 0   temps: --
        Summary at:   Thu Aug  5 02:33:46 EDT 2021

40.0%  proc'd: 835 (17087 Gflop/s)   errors: 0   temps: --
        Summary at:   Thu Aug  5 02:33:47 EDT 2021

60.0%  proc'd: 1670 (16978 Gflop/s)   errors: 0   temps: --
        Summary at:   Thu Aug  5 02:33:48 EDT 2021

80.0%  proc'd: 2672 (16986 Gflop/s)   errors: 0   temps: --
        Summary at:   Thu Aug  5 02:33:49 EDT 2021

100.0%  proc'd: 3674 (16985 Gflop/s)   errors: 0   temps: --
        Summary at:   Thu Aug  5 02:33:50 EDT 2021

100.0%  proc'd: 4676 (16787 Gflop/s)   errors: 0   temps: --
Killing processes.. Freed memory for dev 0
Uninitted cublas
done

Tested 1 GPUs:
        GPU 0: OK
[root@a100_000081005_4-20C_1 bin]#
```

CUDA 测试正常。

4.4. 总结

基于 SR-IOV 的 vGPU 和之前非 SR-IOV 的配置区别主要体现在：1. 对 SR-IOV 硬件和 Kernel 参数的使能。2. 必须使能 GPU 的 SR-IOV 配置。3. 一个 vGPU 实例对应且仅对应一个 GPU VF 设备。4. VM 的设备添加、驱动程序安装以及使用与非 SR-IOV 方式相同。不限于 vCS 类型的 vGPU，也适用于 vPC 和 vWS。